Homework 1: Weather Data Analyzer

Objective:

This assignment involves analyzing weather data from a provided text file (`weather_data.txt`) to calculate various statistics such as average temperature, total rainfall, and more.

Description:

You are provided with a text file named `weather_data.txt`, which contains daily weather data in the following format:

```
Date,Temperature,Rainfall
2025-01-01,32,0.1
2025-01-02,30,0.0
2025-01-03,35,0.2
...
```

Each line represents the date, temperature (in Fahrenheit), and rainfall (in inches) for a day. Your task is to analyze this data using Python by implementing a set of functions in a separate module and using them in a main program.

Requirements:

You will create two files:

1. `weather_analysis.py`: Contains all the necessary functions for analyzing the weather data.
2. `main.py`: The main program that imports `weather_analysis.py` and produces the final results.

---

1. `weather_analysis.py`

This module must contain the following functions:

1. `read_weather_data(file_path)`
    - **Description**: Reads weather data from the provided text file and returns it as a list of tuples.
    - **Parameters**:
        - `file_path (str)`: The path to the weather data file.
    - **Returns**:
        - `List[Tuple(str, float, float)]`: A list of tuples where each tuple contains the date (str), temperature (float), and rainfall (float).
    - **Example**:

```
weather_data = read_weather_data("weather_data.txt")
print(weather_data)
# Output: [('2025-01-01', 32.0, 0.1), ('2025-01-02', 30.0, 0.0)]
```

2. `calculate_average_temperature(data)`

   - **Description**: Calculates and returns the average temperature.
   - **Parameters**:
     - `data (List[Tuple(str, float, float)])`: The weather data.
   - **Returns**:
     - `float`: The average temperature.
   - **Example**:

```
average_temperature = calculate_average_temperature([('2025-01-01',
32.0, 0.1), ('2025-01-02', 30.0, 0.0)])
print(average_temperature)
# Output: 31.0
```

3. `calculate_total_rainfall(data)`

   - **Description**: Calculates and returns the total rainfall.
   - **Parameters**:
     - `data (List[Tuple(str, float, float)])`: The weather data.
   - **Returns**:
     - `float`: The total rainfall.
   - **Example**:

```
total_rainfall = calculate_total_rainfall([('2025-01-01', 32.0, 0.1),
('2025-01-02', 30.0, 0.0)])
print(total_rainfall)
# Output: 0.1
```

4. `find_highest_temperature(data)`

   - **Description**: Finds the highest temperature and its date.
   - **Parameters**:
     - `data (List[Tuple(str, float, float)])`: The weather data.
   - **Returns**:
     - `Tuple[str, float]`: A tuple containing the date and temperature.
   - **Example**:

```
highest_temperature = find_highest_temperature([('2025-01-01', 32.0,
0.1), ('2025-01-03', 35.0, 0.2)])
```

```
print(highest_temperature)
# Output: ('2025-01-03', 35.0)
```

5. `find_lowest_temperature(data)`
   - **Description**: Finds the lowest temperature and its date.
   - **Parameters**:
     - `data (List[Tuple(str, float, float)])`: The weather data.
   - **Returns**:
     - `Tuple[str, float]`: A tuple containing the date and temperature.
   - **Example**:

```
lowest_temperature = find_lowest_temperature([('2025-01-01', 32.0,
0.1), ('2025-01-03', 28.0, 0.2)])
print(lowest_temperature)
# Output: ('2025-01-03', 28.0)
```

6. `find_day_with_most_rainfall(data)`
   - **Description**: Finds the day with the most rainfall and its value.
   - **Parameters**:
     - `data (List[Tuple(str, float, float)])`: The weather data.
   - **Returns**:
     - `Tuple[str, float]`: A tuple containing the date and rainfall.
   - **Example**:

```
most_rainfall = find_day_with_most_rainfall([('2025-01-01', 32.0, 0.1),
('2025-01-03', 28.0, 0.4)])
print(most_rainfall)
# Output: ('2025-01-03', 0.4)
```

---

2. `main.py`

This is the main program that imports the `weather_analysis` module and uses its functions to analyze the data.

This module must contain the following functions:

1. `weather_analyzer(file_path)`
   - **Description:** Analyzes the weather data and returns a dictionary with statistical insights.
   - **Parameters:**
     - `file_path` (str): The path to the weather data file.
   - **Returns:**
     - `dict`: A dictionary containing the calculated statistics.
   - **Example Input:**

```
file_path = "weather_data.txt"
results = weather_analyzer(file_path)
print(results)
```

- **Example Output:**

```
{'average_temperature': 32.33,
 'total_rainfall': 0.3,
 'highest_temperature': {'date': '2025-01-03', 'temperature': 35.0},
 'lowest_temperature': {'date': '2025-01-02', 'temperature': 30.0},
 'most_rainfall': {'date': '2025-01-03', 'rainfall': 0.2}}
```

Import

We do not allow imports on any homework or lab assignments, except for those explicitly specified by that assignment.

On this assignment, you should not import any modules except modules you write yourself, like `weather_analysis.py`.

Grading

For this assignment, most functionality will be auto-graded. We will manually grade for structure and readability. Ensure that you provide good:

- names (for variables and functions)
- whitespace (spacing within a line, and spacing between lines)
- docstrings.

Docstrings in Python

A docstring is a special string used to describe the purpose and functionality of a function, method, class, or module in Python. Docstrings are enclosed in triple quotes and provide documentation that can be accessed using the built-in `help()` function.

Example:

```
def add_numbers(a, b):
    """
    This function adds two numbers and returns the result.

    Parameters:
    a (int or float): The first number.
    b (int or float): The second number.

    Returns:
```

```
    int or float: The sum of the two numbers.
    """
    return a + b
```

Including proper docstrings is mandatory for this assignment and all future homework assignments. Your code will be graded based on the completeness and clarity of your documentation.

Submit

Submit the following files:

- `weather_analysis.py`
- `main.py`

Students must submit to Gradescope individually within 24 hours of the due date (homework due dates are typically Tuesday at 11:59 pm EST) to receive credit.